ulm university universität
uulm

Inst. für Angew. Informationsverarbeitung
Prof. Dr. Franz Schweiggert
Michaela Weiss
Wolfgang Kaifler

16.11.2010
Blatt 4

Systemnahe Software I (WS 2010/2011)

Abgabetermin: 23.11.2010

Aufgabe 1: Theorie (5 Punkte)

- Erklären Sie den Unterschied zwischen den Prä- und Postfix-Operatoren ++ bzw. - - an folgendem Beispiel:

```
1    int i = 1;
2    j = i++;
3    j = ++i;
```

Quellcode 1: Präfix vs. Postfix

- Welche der folgenden Zuweisungen sind syntaktisch und semantisch korrekt und wie sieht die Variablenbelegung nach der Ausführung aller korrekten Zuweisungen (falsche Zuweisungen werden ersatzlos gestrichen) aus?

```
4    int a[] = {1, 2, 3, 4};
5    int b[] = {5, 6, 7, 8};
6    const int c = 3;
7
8    a = b;
9    a[0] = *b;
10   (a[1] += b[1]) += c;
11   c = a[2] + b[2];
12   b[3] += c;
```

Quellcode 2: Zuweisungen

Aufgabe 2: Rechtemanipulation (10 Punkte)

Bei systemnaher Programmierung werden oftmals Bit-Operationen verwendet. So werden beispielsweise bestimmte Datei-Eigenschaften in 16 Bit kodiert und über gezielte Manipulationen von Bits beeinflusst. Die Eigenschaften und zugehörigen Bits sind in Abbildung 1 dargestellt.
Ein wichtiger Anwendungsfall für Bit-Operationen ist das Rechtesystem, das bei UNIX verwendet

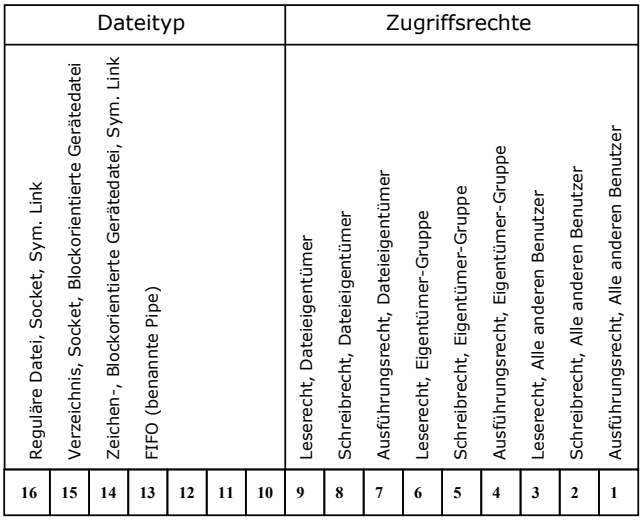

Abbildung 1: Bit-Belegung

und in den niedrigwertigen Bits 1 bis 9 gespeichert wird. Hierzu werden für jede Datei für den „Eigentümer“, die „Eigentümer-Gruppe“ und „alle anderen“ 3 verschiedene Rechte gesetzt. Diese sind Lesen, Schreiben und Ausführen.

Schreiben Sie ein Programm, welches eine 16-Bit-Integer (short int) von der Standardeingabe einliest. Danach soll das Programm sicherstellen, dass alle Rechte gesetzt sind, damit der „Eigentümer“ lesen und schreiben darf. Zusätzlich soll gewährleistet werden, dass sowohl die „Eigentümer-Gruppe“ als auch „alle anderen“ keine Schreibrechte besitzen. Geben Sie den neuen Rechte-Modus aus. Geben Sie des Weiteren aus, ob der „Eigentümer“ die Datei ausführen darf.

Beispiel 1:
Wurde die Zahl 508 eingelesen, so zeigt die Binärdarstellung (111111100), dass der „Eigentümer“ bereits alle Rechte besitzt, die „Eigentümer-Gruppe“ jedoch auch. „Alle anderen“ besitzen derzeit nur das Leserecht. Daher muss der „Eigentümer-Gruppe“ das Schreibrecht genommen werden.
–> **492 (Eigentümer darf Datei ausführen)**

Beispiel 2:
Wurde die Zahl 322 (101000010) eingelesen, besitzt der „Eigentümer“ Lese- und Ausführungsrechte. Ihm muss noch das Schreibrecht gewährt werden. Die „Eigentümer-Gruppe“ hat keine Rechte, somit muss hier nichts verändert werden. „Alle anderen“ haben derzeit Schreibrechte, die entfernt werden müssen.
–> **448 (Eigentümer darf Datei ausführen)**

Aufgabe 3: Tarnung (15 Punkte)

Die Steganographie ist eine Themengebiet, das sich mit der sicheren Speicherung bzw. Übermittlung von Daten beschäftigt. Sie basiert auf der Grundlage, dass einem potentiellen Angreifer gar nicht auffällt, dass die vorliegenden Daten verschlüsselt sind. Ein Beispiel hierfür ist die Verschlüsselung von Text in Zahlenkollonen, der dann z.B. als harmlose Tabellenkalkulation etc. weitergegeben werden kann. So kann man beispielsweise immer 4 Buchstaben eines Textes (ASCII) als long int verpacken. Dies wird nachfolgend beispielhaft verdeutlicht.

Implementieren Sie ein Programm, das diese Tarnung wieder rückgängig macht und entschlüsseln Sie die Datei kalkulation.xls, welche auf der Vorlesungshomepage zum Download bereit steht.

Beispiel: Der Text „try this“ soll getarnt werden.
t = 01110100
00000000 00000000 00000000 01110100

r = 01110010
00000000 00000000 01110100 01110010

y = 01111001
00000000 01110100 01110010 01111001

Leertaste = 00100000
1110100 01110010 01111001 00100000
–> 1953659168

t = 01110100
00000000 00000000 00000000 01110100

h = 01101000
00000000 00000000 01110100 01101000

i = 01101001
00000000 01110100 01101000 01101001

s = 01110011
01110100 01101000 01101001 01110011
–> 1952999795

getarnt: **1953659168 1952999795**

Hinweise:

- Wenn das Dateiende einer einzulesenden Datei erreicht ist, gibt scanf den Wert des Makros **EOF** zurück.
- Wird ein größerer Datentyp in einen kleineren Datentyp gecastet, so bleiben nur die geringerwertigen Bits erhalten, die in den neuen Datentyp passen. Die höherwertigen überzähligen Bits werden verworfen.
  Bsp.:
  unsigned long int i = 4294945450; //11111111 11111111 10101010 10101010
  unsigned int j = (unsigned int) i; //10101010 10101010 (43690)
- In Unix-Systemen ermöglicht das Kommando **wget** den direkten Download einer Datei.
  Bsp.: wget http://www.mathematik.uni-ulm.de/sai/ws10/soft/uebungen/blatt4/kalkulation.xls

Viel Erfolg!